Can a Laplace PDE Define Air Corridors through Low-Altitude Airspace?

Aeris El Asslouj[1], Ella Atkins[2], and Hossein Rastgoftar[1,3]

Abstract— **Urban Uncrewed Aircraft System (UAS) flight will require new regulations that assure safety and accommodate unprecedented traffic density levels. Multi-UAS coordination is essential to both objectives. This paper models UAS coordination as an ideal fluid flow with a stream field governed by the Laplace partial differential equation. Streamlines spatially define closely-spaced deconflicted routes through the airspace and define air corridors that safely wrap buildings and other structures so UAS can avoid collision even when flying among low-altitude vertical obstacles and near mountainous terrain. We divide a city into zones, with each zone having its own sub-network, to allow for modularity and assure computation time for route generation is linear as a function of total area. We demonstrate the strength of our proposed approach by computing air corridors through low altitude airspace of select cities with tall buildings. For US cities, we use open LiDAR elevation data to determine surface elevation maps. We select non-US cities with existing high-fidelity three-dimensional landscape models.**

I. INTRODUCTION

The number of Uncrewed Aircraft Systems (UAS) flying over and within urban regions is projected to rapidly increase [1] for applications such as package delivery [2], aerial photography [3], and airborne sensing [4]. For cities to safely accommodate all new UAS operations, higher air traffic densities, dynamic routing, and UAS-to-UAS coordination must all be supported. UAS will likely operate primarily at low altitudes despite a complex urban landscape to assure separation from new Advanced Air Mobility (AAM) traffic and to maximize efficiency in short-hop (< 2 km) flights.

Increased aerial traffic densities suggest an increased risk of collisions. Apart from Automatic Dependent Surveillance Broadcast (ADS-B), which does not scale, there is currently no capability for UAS-to-UAS detect-and-avoid (DAA) or separation assurance through cooperative routing. In fact, regulators to-date have placed drone coordination DAA in the hand of safety pilots [5] that either visually deconflict in line-of-sight operations or communicate with air traffic control by voice.

Scalable aircraft coordination requires organizational infrastructure including system-wide and vehicle-to-vehicle (V2V) datalink capable of coordinating and sharing vehicle intent (flight route). The UAS community currently envisions modest density distinct flights through sequential predefined corridors as a direct extension of today's manned airport traffic area flight routes. However, even with altitude layers, single-queue fixed corridors impose low traffic density constraints and support limited routing flexibility compared to the cooperative and deconflicted dynamic traffic routing capability proposed in this work.

We define an `air network` as an aerial analog of the multi-lane road network except that the air network is virtual so it can be dynamically reconfigured based on traffic routing requests, winds, and other factors. Through datalink each UAS can update its dynamic air network map in real-time. UAS can also store default air network structures for usage in an offline (pre-flight) mode.

Air networks have two uses. First, they provide [closely-spaced] smooth paths that offer appropriate clearance from buildings, terrain, other structures, and neighboring UAS. The consideration environmental maps and safe separation distances significantly simplifies UAS flight planning and dynamic rerouting. Second, they define safely separated candidate paths that can assure collision avoidance so long as each UAS tracks its planned trajectory to within expected error bounds.

The authors previously presented a method to automatically generate a dense air network for any urban environment that wraps structures [6]. This method takes as input surface elevation data including buildings and other structures and uses the fluid-inspired method presented in [7] to generate layers of air corridors. Each layer is a fixed-altitude plane with air corridors oriented in a fixed nominal direction that wrap around buildings and other structures. This paper presents several advancements to the authors' previous work [6] which increases its scalabity and usability. More specifically, this paper offers the following distinct and novel contributions:

1) The acquisition method for surface elevation data has been fully automated for US cities. In our previous work [6], three dimensional models for buildings were manually created using Google Maps data which is not scalable. We have now built software to automatically query the United States Geological Survey (USGS)'s databases for surface elevation data in the form of point clouds [8]. These point clouds are then processed to create high-resolution surface elevation maps. For cities outside the US, we use existing high-resolution 3d models of cities where this data is available to directly obtain constant-altitude "slices" of buildings and other structures.

[1]Electrical and Computer Engineering Department, University of Arizona, Tucson, Arizona, USA `aymaneelasslouj@arizona.edu`

[2]Aerospace and Ocean Engineering Department, Viginia Tech, Blacksburg, Virginia, USA 24061 `ematkins@vt.edu`

[3]Aerospace and Mechanical Engineering Department, University of Arizona, Tucson, Arizona, USA `hrastgoftar@arizona.edu`

2) We maximize airspace usability by allowing air corridors in the urban canyons between buildings when sufficient clearance exists. This was not the case in our previous work [6] which could only handle wrapping air corridors around a single building.

3) We guarantee a minimum air corridor width by representing air corridors as chains of fixed-size cubes with dimensions set to assure sufficient route separation. In our previous work [6], air corridors were defined as the surface between two streamlines with no restrictions on inter-streamline distance. In this paper, we incrementally build corridors with chains of cubes approximating streamlines while checking at each step that the cubes do not intersect.

4) We split the air network into connected zones where the sub-network of each zone can be generated independently. In our previous work [6], the air network was generated all at once by solving a differential equation. This meant that for each modification the whole network had to be regenerated from scratch. This is no longer the case. Our air network zones can be sized to manage computational cost. Solving the differential equation for a city requires a time that scales with the $3/2$ power of the city's area. When the air network is solved done for each zone independently, the computational time scales linearly with city area.

5) We present sample air networks for major cities to verify that air network generation scales properly and generates dense connected networks for real cities. Results also offer the UAS community visual examples of dense low-altitude air network routing options.

This paper is structured as follows. Section II presents our method to generate an elevation map from USGS data or 3D city model data. This section also describes our method to generate an air network from an elevation map. This method is applied to multiple major cities in Section III with a conclusion and discussion of future work in Section IV.

II. METHODOLOGY

This section presents the steps for generating an air network starting from raw elevation data or a 3d city model. Our method to generate air networks involves generating a discrete elevation map, solving a discrete fluid equation, and iteratively building corridors with finite-sized cubes. As such, we first describe a process to formally discretize the airspace in Section II-A. Then we use either USGS LiDAR elevation data or a 3d model to generate a discrete elevation map in Section II-B. This map is leveraged in Section II-C to solve the ideal fluid flow equation for constant-elevation slices to obtain streamlines that wrap buildings. Finally, streamline maps are used to generate air corridors in Section II-D.

A. Discretization of the airspace

As all of the methods used in our air network generation are discrete, we first must discretize the airspace. The airspace is originally a 3-dimensional continuous space. It

has a spatial origin point $\mathbf{P}_0 = (0,0,0)$ with a reference basis $(\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}})$.
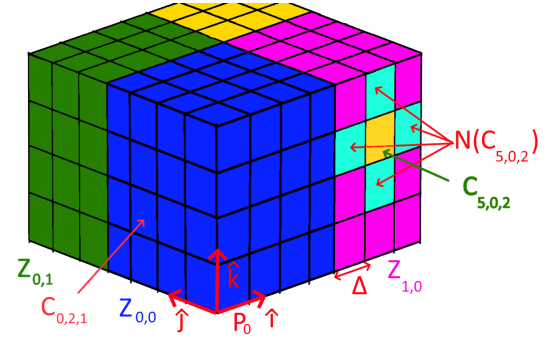

Fig. 1: Example cell discretization with anchor point $\mathbf{P}_0$, reference frame $(\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}})$, cell side size Δ, and zone base side length $N = 4$. Zones $Z_{0,0}$, $Z_{0,1}$, and $Z_{1,0}$ are shown respectively in blue, green, and pink. Cell $C_{5,0,2}$ is shown in yellow and its neighbors $\mathbf{N}(C_{5,0,2})$ are shown in cyan.

To discretize it, we cover the airspace with a 3-dimensional grid of cubic cells as shown in Fig. 1. The axes of the grid align with the basis vectors and each cell has side length $\Delta \in \mathbb{R}^+$. We assign to each cell discrete grid coordinates $(i,j,k) \in \mathbb{Z}^3$ with each cell being referenced as $C_{i,j,k}$. The set of all cells is denoted $\mathbf{C}$. Each cell encloses a given cubic region of the originally continuous airspace. Given a point in the airspace, we can determine the cell that encloses it by getting the cell's grid coordinates using the function $\mathbf{F}$:

$$
\begin{aligned}
\mathbf{F}: \quad \mathbb{R}^3 \quad &\rightarrow \quad \mathbb{Z}^3 \\
(x,y,z) \quad &\rightarrow \left(\left\lfloor \frac{x}{\Delta} \right\rfloor, \left\lfloor \frac{y}{\Delta} \right\rfloor, \left\lfloor \frac{z}{\Delta} \right\rfloor \right)
\end{aligned}
\tag{1}
$$

where $\lfloor . \rfloor$ is the floor function.

Each cell is considered connected to the cells with which it is in direct physical contact. For this to happen, the cells' grid coordinates must differ in at least one dimension. The cells with which a cell is connected are called its Cartesian neighbors. Sample Cartesian neighbors of a cell are shown in Fig. 1. The set of all Cartesian neighbors of a cell $C \in \mathbf{C}$ is denoted $\mathbf{N}(C)$ and given by function $\mathbf{N}$:

$$
\begin{aligned}
\mathbf{N}: \quad \mathbf{C} \quad &\rightarrow \quad \mathcal{P}(\mathbf{C}) \\
C_{i,j,k} \quad &\rightarrow \{ C_{i+1,j,k}, C_{i-1,j,k}, C_{i,j+1,k}, \\
& \quad C_{i,j-1,k}, C_{i,j,k+1}, C_{i,j,k-1} \}
\end{aligned}
\tag{2}
$$

where power set $\mathcal{P}(\mathbf{C})$ define the set of all subsets of $\mathbf{C}$.

We now have a connected discretized airspace made of a grid of cells. We divide this discretized airspace into zones. This is to allow for the air network to be generated for each zone independently. Each zone consists of multiple cells. A zone has a base of $N \times N$ cells and contains all cells above them. Sample zones are shown in Fig. 1. The set of all zones $\mathbf{Z}$ forms a partition of the set of all cells $\mathbf{C}$. We assign each zone coordinates $(a,b) \in \mathbb{Z}^2$ and refer to it as $\mathbf{Z}_{a,b}$. The set $\overline{\mathbf{Z}_{a,b}}$ contains the (i,j) cell coordinates of the cells that $\mathbf{Z}_{a,b}$ covers. To find all the cells contained in a zone, we use the

following:

$$\overline{\mathbf{Z}_{a,b}} = \left\{ (i,j) \in \mathbb{Z}^2 \middle| \begin{array}{l} Na \le i < N(a+1) \\ Nb \le j < N(b+1) \end{array} \right\}, \forall (a,b) \in \mathbb{Z}^2. \quad \text{(3a)}$$

$$\mathbf{Z}_{a,b} = \left\{ C_{i,j,k} \in \mathbf{C} \middle| (i,j) \in \overline{\mathbf{Z}_{a,b}} \right\}, \quad \forall (a,b) \in \mathbb{Z}^2. \quad \text{(3b)}$$

B. Elevation map generation

Urban airspace contains obstacles in the form of buildings and other structures. In order to generate an air network that safely wraps around all obstacles, we have to identify which cells of the discrete airspace intersect with an obstacle. As such, we split cells into two types. A free cell is a cell that does not intersect any obstacle. A full cell is a cell which does intersect an obstacle.

To simplify our model, we merge all buildings and structures with the ground and consider all of it as a terrain of varying altitude. As such, for each horizontal cell coordinates $(i,j) \in \mathbb{Z}^2$, there is a maximum cell altitude under which all cells are full and above which all cells are free. For each zone $\mathbf{Z}_{a,b} \in \mathbf{Z}$, we define a discrete elevation map $M_{a,b} : \overline{\mathbf{Z}_{a,b}} \to \mathbb{Z}$ which gives this "ground" cell altitude for each horizontal cell coordinates. As such, the type of a cell can be obtained by the function T:

$$\begin{aligned} T: \quad \mathbf{Z}_{a,b} \quad &\to \quad \{\text{full}, \text{free}\} \\ C_{i,j,k} \quad &\to \begin{cases} \text{full} & k \le M_{a,b}(i,j) \\ \text{free} & k > M_{a,b}(i,j) \end{cases}. \end{aligned} \quad \text{(4)}$$

We use two different methods to obtain the discrete elevation map $M_{a,b}$ of a zone. If the city for which we are generating the air network is in the United States, we use the USGS's LiDAR elevation data. If the city is outside the US, we must use an existing 3d model of the city.

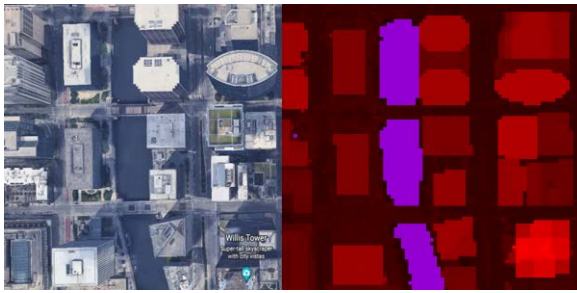*1) Elevation map for US cities from USGS point clouds:*

Fig. 2: Google Earth satellite view of city area in Chicago (left) and its corresponding elevation map based on the USGS's LiDAR data (right). The intensity of red is proportional to the square root of cell altitude. Purple shows (i,j) coordinate pairs with no altitude data where $M_{1,0}$ is positive infinity.

For most regions in the US, the USGS provides LiDAR elevation data [8]. LiDAR point clouds have been obtained from equipped aircraft and rotorcraft overflying an area to estimate terrain elevation including buildings and other

structures. Note that LiDAR may not capture narrow vertical obstacles and power lines; researchers are investigating complementary methods to address this challenge [9]. For a given zone, our system queries the USGS's database for any point cloud with points that are within the zone. The point clouds found are grouped into a set $\mathbf{P}_{a,b}$. The point clouds come with a projected coordinate system. So we first translate all the points such that our origin $\mathbf{P}_0$ coincides with the origin of the data's coordinate system. Then, we map the points to cells using the function $\mathbf{F}$ defined in (1) to get a cell cloud $\mathbf{F}(\mathbf{P}_{a,b})$. Finally, for each horizontal cell coordinate $(i,j) \in \overline{\mathbf{Z}_{a,b}}$, we pick the highest cell altitude from the cell cloud as the terrain altitude for the discrete elevation map $M_{a,b}(i,j)$. Therefore, $M_{a,b}(i,j)$ is given by:

$$\begin{aligned} M_{a,b}: \quad \overline{\mathbf{Z}_{a,b}} \quad &\to \quad \mathbb{Z} \\ (i,j) \quad &\to \max\left\{ k \middle| (i,j,k) \in \mathbf{F}(\mathbf{P}_{a,b}) \right\} \end{aligned} \quad \text{(5)}$$

Certain horizontal coordinates might not have any cells from the cell cloud that match them. In that case, we assign infinity as the terrain altitude for them. This is the most conservative approach as it assumes the worst case scenario of there being an extremely high feature in the space with no elevation data. A sample discrete elevation map is shown in Fig. 2.

2) Elevation map for non-US cities from 3d models:

In the case of non-US cities, we use exiting 3d models as our data source. For these cities, we use the 3d modeling software Blender [10] to take slices of the 3d model at specific cell altitudes. These slices allow to distinguish between "full" and "free" cells at every specified altitudes.

While this method is more straightforward, it assumes the existence of a 3d model for the city which is not guaranteed. It is also for now a manual process, although it is possible to automate the slicing. Lastly, high fidelity 3d models are often paid models as opposed to USGS data which is public, free, and available for about all of the continental US.

C. Streamlines map generation

Our goal is to generate air corridors at fixed-altitude layers that wrap around buildings. For this, we solve the equation of an ideal irrotational fluid flowing in a fixed-altitude 2-dimensional plane in between buildings. This allows us to obtain the streamline distribution in the layer in question. The streamline distribution is used in Section II-D to guide air corridor generation.

The input parameters are the number of layers for the air network $M \in \mathbf{N}$, and for each layer $n \in \{1 \cdots M\}$, its altitude h_n and the nominal direction of its air corridors $\hat{\mathbf{d}}_n$.

For each zone, we have to solve the fluid flow equation at each of the given altitudes. In this section, we fix the zone to be $\mathbf{Z}' \in \mathbf{Z}$ and the layer in question to be $M' \in \{1 \cdots M\}$ such that $h' = h_{M'}$ and $\hat{\mathbf{d}}' = \hat{\mathbf{d}}_{M'}$.

First, we convert the continuous air space altitude h' to a cell altitude k' using the function $\mathbf{F}$ defined in (1). Then we take a horizontal slice $\tilde{\mathbf{Z}}$ of the zone $\mathbf{Z}'$ at cell altitude k'.

A horizontal slice of a zone at cell altitude k is the subset of cells within the zone that have cell altitude k. As such:

$$\tilde{\mathbf{Z}} = \left\{ C_{i,j,k} \in \mathbf{Z}' | k = k' \right\}. \qquad (6)$$

If a continuous ideal irrotational fluid were to flow in the slice $\tilde{\mathbf{Z}}$, it would form streamlines such that each point in the plane would be part of a given streamline. Each streamline can be assigned a real index ψ, and this ψ distribution satisfies Laplace's equation:

$$\nabla^2 \psi = 0. \qquad (7)$$

In our case, the slice is made of discrete cell. So the ψ distribution is approximated with a discrete version which assigns to each cell in the slice $C \in \tilde{\mathbf{Z}}$ a streamline index $\psi(C)$. Laplace's equation also transform to its discrete form:

$$\sum_{C' \in (\mathbf{N}(C) \cap \tilde{\mathbf{Z}})} (\psi(C) - \psi(C')) = 0, \quad \forall C \in \tilde{\mathbf{Z}}. \qquad (8)$$

In Eq. (8), $(\mathbf{N}(C) \cap \tilde{\mathbf{Z}})$ is the set of all neighbors of C which are in the same slice. To obtain the ψ distribution, we first have to define the boundary conditions for the Laplace difference equation, then solve it.

1) Boundary conditions:

There are two types of boundary conditions. The first is the ψ values of boundary cells which control the nominal direction of streamlines. The second is the ψ value of full cells which control how streamlines wrap around them.
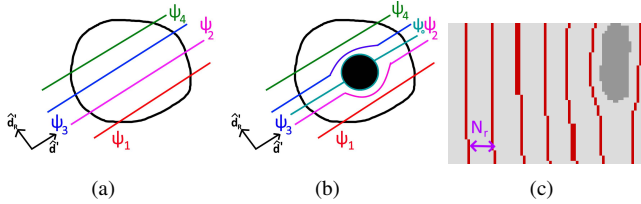

(a)   (b)   (c)

Fig. 3: (a) Visualization of method for assigning ψ values to boundary cells. The nominal streamline direction is $\hat{\mathbf{d}}'$ while $\hat{\mathbf{d}}'_R$ is orthogonal to it. (b) Visualization of method for assigning ψ values to obstacles. The obstacle is assigned ψ_o such that a single streamline passes by its boundary. Here $\psi_4 < \psi_3 < \psi_o < \psi_2 < \psi_1$. (c) Illustration of minimum cell distance between starting points of two air corridors.

The set of all cells on the boundary of the slice is denoted $\partial\tilde{\mathbf{Z}}$. Each two cells on the boundary that have the same ψ value are on the same streamline. Therefore, in order for streamlines to have a nominal direction of $\hat{\mathbf{d}}'$, any two points on a line with direction $\hat{\mathbf{d}}'$ must have have the same ψ value as shown in Fig. 3 (a). One way to achieve this is to have the ψ value of boundary cells change when moving orthogonally to $\hat{\mathbf{d}}'$. The simplest way to do it is to rotate $\hat{\mathbf{d}}'$ by 90 degrees to get $\hat{\mathbf{d}}'_R$, then have the ψ value of a boundary cell be the cell's coordinate along $\hat{\mathbf{d}}'_R$. This yields:

$$\psi(C_{i,j,k}) = \begin{bmatrix} i & j \end{bmatrix}^T . \hat{\mathbf{d}}'_R, \qquad \forall C_{i,j,k} \in \partial\tilde{\mathbf{Z}}, \qquad (9)$$

where "·" is the the inner product symbol.

Eq. (9) is a more general form of the formulas used in [6] to get the ψ value of boundary cells. This is because Eq. (9) works with any nominal streamline direction $\hat{\mathbf{d}}'$ whereas the formulas in [6] only work for $\hat{\mathbf{d}}' = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$ or $\hat{\mathbf{d}}' = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$.

Groups of full cells that are all neighbors of each other form an obstacle. To partition full cells into distinct obstacles, we start with any non-categorized full cell and iteratively group it with full cell neighbors. Once no full cell neighbors are left, the obstacle is complete and the process starts again with another non-categorized full cell. Once all full cells have been categorized into obstacles, we assign their ψ values.

To have streamlines wrap around obstacles as shown in Fig. 3 (b), the boundary cells of obstacles must have the same ψ value. This is equivalent to having all cells of the obstacle taking a single ψ value. Streamlines with a higher ψ will pass around the obstacle from one side, while streamlines with a lower ψ will pass around from the other side.

Our method here differs from our previous work. In our previous paper [6], all obstacles were assigned $\psi = 0$. This made it such that streamlines could not go in between buildings and instead all went by the boundary of the urban region. This is because setting $\psi = 0$ for all obstacles effectively groups them into a single obstacle covered by a single streamline. To fix this issue, we modified our method for assigning ψ values for obstacles.

Our goal is, for each obstacle, to select the ψ value which minimally disturbs the flow of streamlines. As such, we first pick the center coordinate of the obstacle $\mathbf{r}_o$. We then convert it to cell coordinates using the function $\mathbf{F}$. Finally, we use the same formula that we used for boundary cells, Eq. (9), to select its ψ value:

$$\psi_o = \mathbf{F}(\mathbf{r}_o)^T . \hat{\mathbf{d}}'_R. \qquad (10)$$

2) Solving the difference Laplace equation

In this section, we introduce a new indexing system for cells in the slice $\tilde{\mathbf{Z}}$ to simplify our equations. Instead of referring to cells using three indices as in $C_{i,j,k}$, we refer to cells with a single index $I \in \{1, \cdots, m\}$ where $m = N^2$ is the number of cells in the slice. This new indexing system is designed to group cells into the following categories:

1) $(1, \cdots, m_b)$ are boundary cells.
2) $(m_b+1, \cdots, m_b+m_f)$ are free non-boundary cells.
3) $(m_b+m_f+1, \cdots, m_b+m_f+m_o)$ are full non-boundary cells.

with:

1) $\boldsymbol{\psi}_b = \begin{bmatrix} \psi(C_1) & \cdots & \psi(C_{m_b}) \end{bmatrix}^T$.
2) $\boldsymbol{\psi}_f = \begin{bmatrix} \psi(C_{m_b+1}) & \cdots & \psi(C_{m_b+m_f}) \end{bmatrix}^T$.
3) $\boldsymbol{\psi}_o = \begin{bmatrix} \psi(C_{m_b+m_f+1}) & \cdots & \psi(C_{m_b+m_f+m_o}) \end{bmatrix}^T$.

Let $\mathbf{L} = [L_{IJ}] \in \mathbb{R}^{m \times m}$ be the Laplacian matrix [11] for the slice $\tilde{\mathbf{Z}}$ whose entries indicate the number of neighbors each cell has in $\tilde{\mathbf{Z}}$:

$$L_{IJ} = \begin{cases} |\mathbf{N}(C_I) \cap \mathbf{Z}_k| & I = J \\ -1 & I \neq J \wedge C_J \in N(C_I), \\ 0 & \text{otherwise} \end{cases} \qquad (11)$$

where neighboring function $\mathbf{N}(\cdot)$ was defined in Eq. (2).

Matrix $\mathbf{L}$ can be split into nine sub-matrices which define the connections between the three groups of cells:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_{bb} & \mathbf{L}_{bf} & \mathbf{L}_{bo} \\ \mathbf{L}_{fb} & \mathbf{L}_{fc} & \mathbf{L}_{fo} \\ \mathbf{L}_{ob} & \mathbf{L}_{of} & \mathbf{L}_{oo} \end{bmatrix}. \tag{12}$$

With this established, we can combine all instances of the Laplace difference equation Eq. (8) into one linear system:

$$\mathbf{L}_{fb}\boldsymbol{\psi}_b + \mathbf{L}_{fc}\boldsymbol{\psi}_f + \mathbf{L}_{fo}\boldsymbol{\psi}_o = 0. \tag{13}$$

We know the ψ values of all boundary and full non-boundary cells which are $\boldsymbol{\psi}_b$ and $\boldsymbol{\psi}_o$. So we can solve for the ψ values of free non-boundary cells $\boldsymbol{\psi}_f$ with:

$$\boldsymbol{\psi}_f = -\mathbf{L}_{fc}^{-1}(\mathbf{L}_{fb}\boldsymbol{\psi}_b + \mathbf{L}_{fo}\boldsymbol{\psi}_o). \tag{14}$$

This linear system should not be solved with a regular solver. This is because $\mathbf{L}$ is usually very large as its size is $m^2 = N^4$. Also the number of its non-zero entries is in the order of N^2, which makes it a sparse matrix [12]. As such, we using a sparse linear system solving method to solve the set of linear equations provided by Eq. (12) with a low computation cost. Because both $\mathbf{L}$ and $\mathbf{L}_{fc}$ are symmetric, we use the conjugate gradient method [13] to obtain $\boldsymbol{\psi}_f$.

D. Air corridor generation

With a complete discrete streamline distribution ψ, we can finally generate air corridors. Air corridors are chains of connected cells that start and end on the boundary of the slice $\tilde{\mathbf{Z}}$. They also approximate streamlines and do not intersect obstacles or each other.

Our definition of air corridors ensures they have a finite width equal to the size of cells Δ. This differs from previous work [6] where air corridors were defined as the area between two streamlines. The problem with directly using streamlines to get corridor boundaries is that we could not find a straightforward method to ensure streamlines are separated by a minimum distance. This is important as drones have a finite width, so we must ensure that air corridors have a minimum width which is what our new definition of air corridors does.

To generate an air corridor, we start at a boundary cell $C_b \in \partial\tilde{\mathbf{Z}}$. Boundary cells are split into three groups based on a property we call orientation.

1) If at C_b's position, $\hat{\mathbf{d}}'$ points inside the slice, C_b is oriented forward.
2) If at C_b's position, $\hat{\mathbf{d}}'$ points outside the slice, C_b is oriented backwards.
3) Otherwise, the C_b has no orientation.

We only consider C_b as a starting point for a corridor if it has an orientation. If C_b is orientated forward, we expand the corridor in the $\hat{\mathbf{d}} = \hat{\mathbf{d}}'$ direction. If C_b is oriented backwards, we use $\hat{\mathbf{d}} = -\hat{\mathbf{d}}'$ instead. To generate the air corridor, we use algorithm 1.

The main idea behind algorithm 1 is to iteratively add cells while trying to follow the streamline. the air corridor is a list of cells. At each run through the loop, all neighbors of the

---

Algorithm 1: Algorithm to generate a corridor from the ψ distribution.

Data: C_b, $\hat{\mathbf{d}}$, ψ, previously built air corridors
Result: AirCorridor is an air corridor that starts at C_b and goes in the $\hat{\mathbf{d}}$ direction

LastCell $\leftarrow C_b$;
AirCorridor $\leftarrow$ List;
Add LastCell to AirCorridor;
OriginalPsi $\leftarrow \psi(C_b)$;
while *LastCell $\notin \partial\mathbf{Z}_k$ OR Orientation(LastCell) = Orientation(C_b)* **do**
    Neighbors $\leftarrow N(\text{LastCell}) \cap \mathbf{Z}_k$;
    Discard neighbors that are full;
    Discard neighbors that are part of AirCorridor;
    Discard neighbors whose progress along $\hat{\mathbf{d}}$ is smaller than the progress of LastCell;
    (The progress of a cell $C_{i,j,k} \in \mathbf{Z}_k$ along $\hat{\mathbf{d}}$ is defined as $\begin{bmatrix} i & j \end{bmatrix}^T \cdot \hat{\mathbf{d}}$.);
    if *No neighbors remain* **then**
        Stop, attempt failed;
    end
    NextCell $\leftarrow$ neighbor whose ψ is the closest to OriginalPsi;
    if *NextCell is part of another corridor* **then**
        Stop, attempt failed;
    end
    Add NextCell to AirCorridor;
    LastCell $\leftarrow$ NextCell;
end
Attempt successful;

---

last cell in the corridor are selected. From the selected cells, we remove the cells that are full, already part of the corridor, or that would not advance the corridor along the direction of expansion $\hat{\mathbf{d}}$. This is to ensure the corridor does not contain obstacle cells and does not loop too much. If none of them remain, then a dead end was reached and the corridor is discarded. Otherwise, we pick the neighbor with the closest ψ value to that of the original cell C_b. This is to ensure the corridor stays near the same streamline. If the selected cell is already part of established corridor, it is considered a dead end and the corridor is discarded. This is to avoid corridor intersections and excessive deviations. If the selected cell is available, it is added to the corridor and the loop runs again. The loop stops when the corridor reaches a boundary cell with a different orientation from C_b, meaning it reached the other end of the slice boundary.

Air corridor generation is attempted at each cell on the boundary of the slice. The zone for which we are generating these air corridors might have neighbor zones with existing sub-networks. If that is the case, we give priority to the boundary cells which would allow connections with the neighbor zones' existing corridors. When all of these potential connections are attempted, then we try the rest

of the boundary cells. This ensures maximum inter-zone connectivity for the air network.

To avoid high variations in the network density, we add a minimum cell distance N_r between boundary cells at which we attempt to generate corridors (See Fig. 3(c)).

Within the zone $\mathbf{Z}'$, for each two vertically adjacent layers of corridors, we allow vertical connections between them. These vertical connections are in the form of vertical chains of cells which connect the corridors of each of the two layers. They happen when cells with the same horizontal coordinate (i, j) within each slice are both part of a corridor in their respective layer. This ensures maximum inter-layer connectivity for the air network.

III. RESULTS

We tested our air network generation method on sections of the cities listed in Table I. Table I also includes the corresponding air network parameters table for each city and the time it took to generate them. For Dubai, Shanghai, and Tokyo, we used existing 3d models. For Chicago, we used USGS LiDAR elevation data to create their discrete elevation maps. Pictures of the 3d models for ubai, Shanghai, and Tokyo are shown in Fig. 4(a), Fig. 4(b), and Fig. 4(c) respectively. These figures also show the constant-altitude slices made for the air network layers. The discrete elevation maps $M_{1,0}$ of the zone $\mathbf{Z}_{1,0}$ for Chicago is shown in Fig. 4(d).

For all cities, the air network was made to have two layers with orthogonal nominal flow directions. The lower level flows in the +x direction while the upper level flows in the +y direction. The generated air networks for Dubai, Shanghai, and Tokyo are shown in Fig. 5 (a), Fig. 5 (b), and Fig. 5 (c) respectively. As seen in Figs. 5 (a-c), the generated air network is composed of two zones $\mathbf{Z}_{0,0}$ and $\mathbf{Z}_{0,1}$. Figure 5 (d) shows the upper layer of Dubai's air network which show cases that air corridors do not intersect (The upper layer of Dubai's air network demonstrates that our method successfully generates inter-zone connections for a high percentage of corridors). Furthermore, the air network generated for Chicago is shown in Fig. 6. We notice that the air network figures (Figs. 5 (a-c) and Fig. 6) show the networks as seen from above, so the air corridors appear to intersect. This is not the case since corridors flowing in +x and corridors flowing in +y are in two different layers separated vertically.

TABLE I: Cities used to get sample air networks

City, Country	Parameters	Generation time
Dubai, United Arab Emirates	Table II	20.0 s
Shanghai, China	Table III	3.63 s
Tokyo, Japan	Table IV	3.00 s
Chicago, United States	Table V	1.78 s

IV. CONCLUSION

This paper presented an an air network generation method with potential to safely support high UAS traffic densities

TABLE II: Parameters of air network for Dubai

Name	Value
Anchor $\mathbf{P}_0$ longitude	lon = 55.251389°
Anchor $\mathbf{P}_0$ latitude	lat = 25.187778°
Anchor $\mathbf{P}_0$ AMSL altitude	alt = 2 meters
Coordinate system	Custom
Cell size	Δ = 5 meters
Zone horizontal size	N = 476
First corridor layer AMSL altitude	h_1 = 80 meters
First corridor layer direction	$\hat{\mathbf{d}}_1 = \begin{bmatrix} 1.0 & 0.0 \end{bmatrix}$
Second corridor layer AMSL altitude	h_2 = 90 meters
Second corridor layer direction	$\hat{\mathbf{d}}_2 = \begin{bmatrix} 0.0 & 1.0 \end{bmatrix}$
Minimum cells between ends of corridors	N_r = 10

TABLE III: Parameters of air network for Shanghai

Name	Value
Anchor $\mathbf{P}_0$ longitude	lon = 121.496944°
Anchor $\mathbf{P}_0$ latitude	lat = 31.22889°
Anchor $\mathbf{P}_0$ AMSL altitude	alt = 0 meters
Coordinate system	Custom
Cell size	Δ = 5 meters
Zone horizontal size	N = 350
First corridor layer AMSL altitude	h_1 = 99 meters
First corridor layer direction	$\hat{\mathbf{d}}_1 = \begin{bmatrix} 1.0 & 0.0 \end{bmatrix}$
Second corridor layer AMSL altitude	h_2 = 129 meters
Second corridor layer direction	$\hat{\mathbf{d}}_2 = \begin{bmatrix} 0.0 & 1.0 \end{bmatrix}$
Minimum cells between ends of corridors	N_r = 10

TABLE IV: Parameters of air network for Tokyo

Name	Value
Anchor $\mathbf{P}_0$ longitude	lon = 139.776944°
Anchor $\mathbf{P}_0$ latitude	lat = 35.675278°
Anchor $\mathbf{P}_0$ AMSL altitude	alt = 4 meters
Coordinate system	Custom
Cell size	Δ = 5 meters
Zone horizontal size	N = 287
First corridor layer AMSL altitude	h_1 = 76 meters
First corridor layer direction	$\hat{\mathbf{d}}_1 = \begin{bmatrix} 1.0 & 0.0 \end{bmatrix}$
Second corridor layer AMSL altitude	h_2 = 106 meters
Second corridor layer direction	$\hat{\mathbf{d}}_2 = \begin{bmatrix} 0.0 & 1.0 \end{bmatrix}$
Minimum cells between ends of corridors	N_r = 10

TABLE V: Parameters of air network for Chicago

| Name	Value
Anchor $\mathbf{P}_0$ longitude	lon = -87.65250°
Anchor $\mathbf{P}_0$ latitude	lat = 41.87833°
Anchor $\mathbf{P}_0$ AMSL altitude	alt = 4 meters
Coordinate system	EPSG 6455
Cell size	Δ = 5 meters
Zone horizontal size	N = 200
First corridor layer AMSL altitude	h_1 = 235 meters
First corridor layer direction	$\hat{\mathbf{d}}_1 = \begin{bmatrix} 1.0 & 0.0 \end{bmatrix}$
Second corridor layer AMSL altitude	h_2 = 245 meters
Second corridor layer direction	$\hat{\mathbf{d}}_2 = \begin{bmatrix} 0.0 & 1.0 \end{bmatrix}$
Minimum cells between ends of corridors	N_r = 10
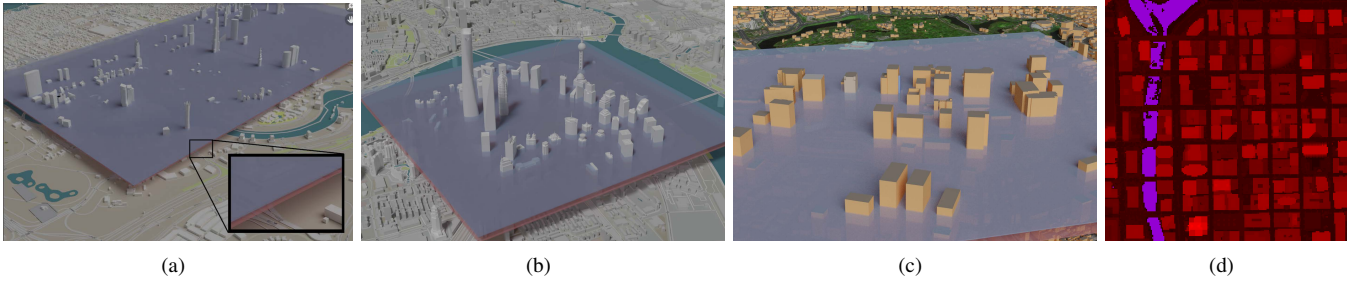
Fig. 4: 3d models used for (a) Dubai [14], Shanghai [15], and Tokyo [16]. In these three sub-figures, the slice for the lower layer of the air network is shown in red while the slice for the upper layer of the air network is shown in blue. (d) Discrete elevation map $M_{1,0}$ of the zone $\mathbf{Z}_{1,0}$ for Chicago based on the USGS's LiDAR data. The intensity of red is proportional to the square root of cell altitude. Purple shows (i, j) coordinate pairs with no altitude data where $M_{1,0}$ is positive infinity.
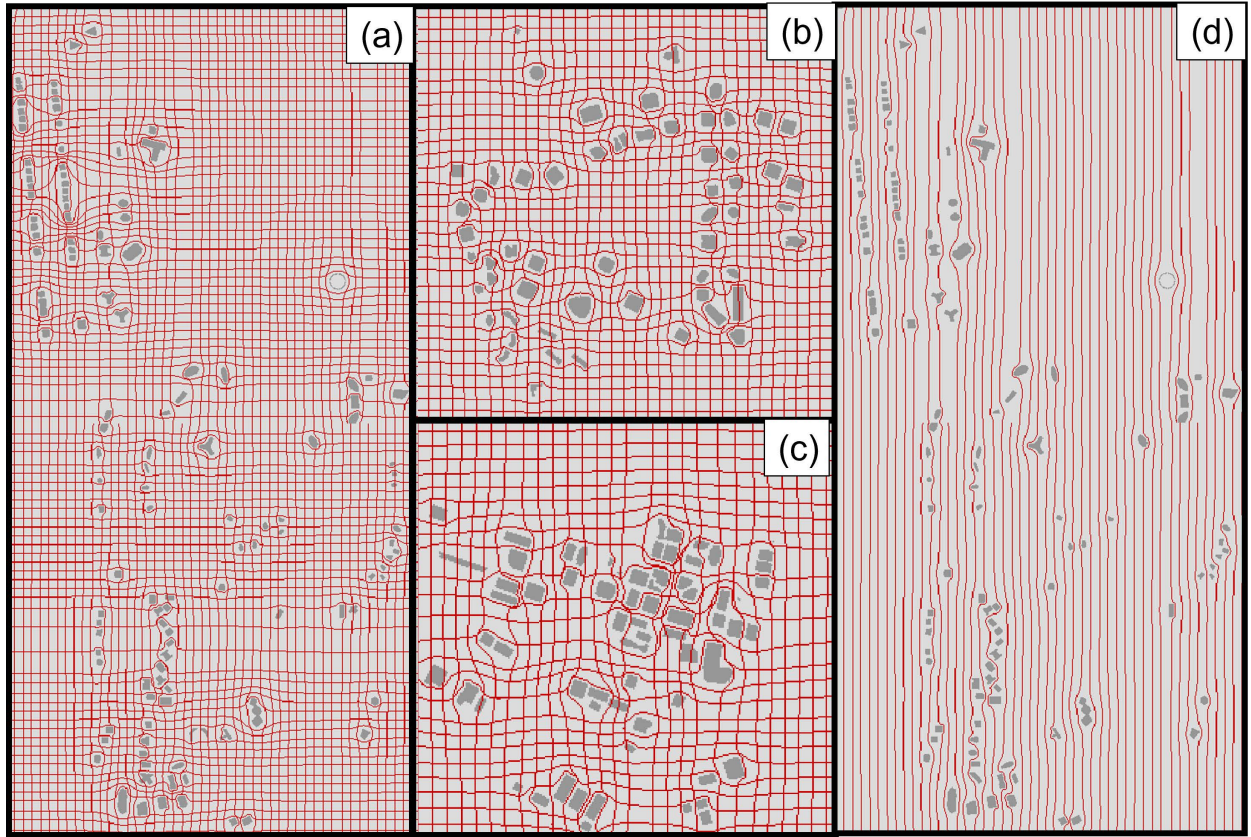

Fig. 5: Air network for (a) Dubai, (b) Shanghai, and (Tokyo) through zones $\mathbf{Z}_{0,0}$ (bottom) and $\mathbf{Z}_{0,1}$ (top). In these sub-figures (a-c), red shows corridor cells in both layers while grey shows full cells through the slice at the lower layer's altitude. (d) Upper layer of air network for Dubai through zones $\mathbf{Z}_{0,0}$ (bottom) and $\mathbf{Z}_{0,1}$ (top). Red shows corridor cells while grey shows full cells through the slice at the upper layer's altitude.

for low-altitude flights through dense urban regions. This work improves on the authors' previous work in terms of automation, scalability, safety of route separation, and computational overhead. Fluid flow equations used to define streamlines between buildings was solved using a sparse linear system solver and solved independently for each zone of the city to minimize computation time. Air corridors were guaranteed to have a minimum width through the use of iterative generation with finite-sized blocks.

Future research should focus on improving the process more by accounting for variance in altitude. Impact of air network parameters such as zone size, number of layers, and density should also be further examined in the context of metrics including but not limited to safe separation, traffic density, datalink delay, and realistic UAS trajectory tracking accuracies. Future work might also examine how this work can evolve to safely and efficiently support UAS with different performance characteristics operating in the same urban air network.
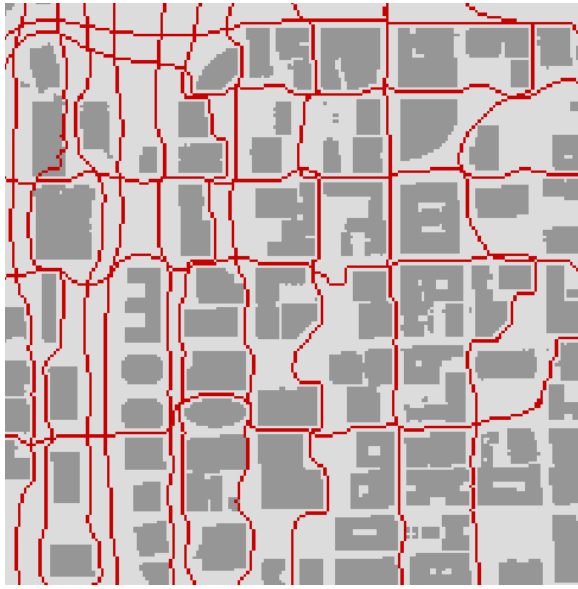
Fig. 6: Air network for Chicago through zones $\mathbf{Z}_{0,0}$ (bottom) and $\mathbf{Z}_{0,1}$ (top). Red shows corridor cells in both layers while grey shows full cells through the slice at the lower layer's altitude.

REFERENCES

[1] Mordor Intelligence Industry Reports, "Unmanned aerial vehicles market - growth, trends, covid-19 impact, and forecast (2022 - 2027)," Mordor Intelligence.

[2] S. Jung and H. Kim, "Analysis of amazon prime air uav delivery service," *Journal of Knowledge Information Technology and Systems*, vol. 12, no. 2, pp. 253–266, 2017.

[3] G. Eckert, S. Cassidy, N. Tian, and M. E. Shabana, "Using aerial drone photography to construct 3d models of real world objects in an effort to decrease response time and repair costs following natural disasters," in *Science and Information Conference*. Springer, 2019, pp. 317–325.

[4] A. V. Savkin and H. Huang, "A method for optimized deployment of a network of surveillance aerial drones," *IEEE Systems Journal*, vol. 13, no. 4, pp. 4474–4477, 2019.

[5] "Recreational flyers & community-based organizations." [Online]. Available: https://www.faa.gov/uas/recreational_flyers

[6] H. Emadi, E. Atkins, and H. Rastgoftar, "A finite-state fixed-corridor model for uas traffic management," *arXiv preprint arXiv:2204.05517*, 2022.

[7] H. Rastgoftar and E. Atkins, "Physics-based freely scalable continuum deformation for uas traffic coordination," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 2, pp. 532–544, 2020.

[8] "The national map." [Online]. Available: https://www.usgs.gov/programs/national-geospatial-program/national-map

[9] P. R. Flanigen, R. R. Copeland, N. Sarter, and E. M. Atkins, "Current challenges and mitigations for airborne detection of vertical obstacles," in *Virtual, Augmented, and Mixed Reality (XR) Technology for Multi-Domain Operations III*, vol. 12125. SPIE, 2022, pp. 57–69.

[10] B. Foundation, "Home of the blender project - free and open 3d creation software." [Online]. Available: https://www.blender.org/

[11] R. Merris, "Laplacian matrices of graphs: a survey," *Linear Algebra and its Applications*, vol. 197-198, pp. 143–176, 1994. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0024379594904863

[12] T. A. Davis, S. Rajamanickam, and W. M. Sid-Lakhdar, "A survey of direct methods for sparse linear systems," *Acta Numerica*, vol. 25, p. 383–566, 2016.

[13] J. L. Nazareth, "Conjugate gradient method," *WIREs Computational Statistics*, vol. 1, no. 3, pp. 348–353, 2009. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.13

[14] "Dubai business bay uae: 3d model." [Online]. Available: https://www.cgtrader.com/3d-models/exterior/cityscape/dubai-business-bay-uae-5km

[15] "Shanghai downtown china: 3d model." [Online]. Available: https://www.cgtrader.com/3d-models/exterior/cityscape/shanghai-downtown-china

[16] "Cityscape tokyo japan: 3d model." [Online]. Available: https://www.cgtrader.com/3d-models/exterior/cityscape/cityscape-tokyo-japan-7caab6ac-2b3c-4de0-82a0-7a30a7e29184